Wide and Deep in Multi-factor Stock Selection

by Jiale Hu and Zhaopu Wang with the help of Xiang Ao and Feiyang Pan

1. INTRODUCTION

In 2016, Google introduced an algorithm named *wide and deep learning* which combines linear regression and deep learning together. It is known that linear regression is good at fitting a model but lack in generalization and interactions between each factor should always be considered. In comparison, deep learning has better performance at generalization, and interactions can be learned automatically. However, the neural network doesn't work as well as linear regression at fitting a model. When combining LR and neural network together, the ability of fitting and generalizing a model can be boosted. Google initially tends to use this algorithm on recommender systems, in this paper, we are going to investigate if *wide and deep learning* has better performance than linear regression on multi-factor stock selections in Chinese stock market.

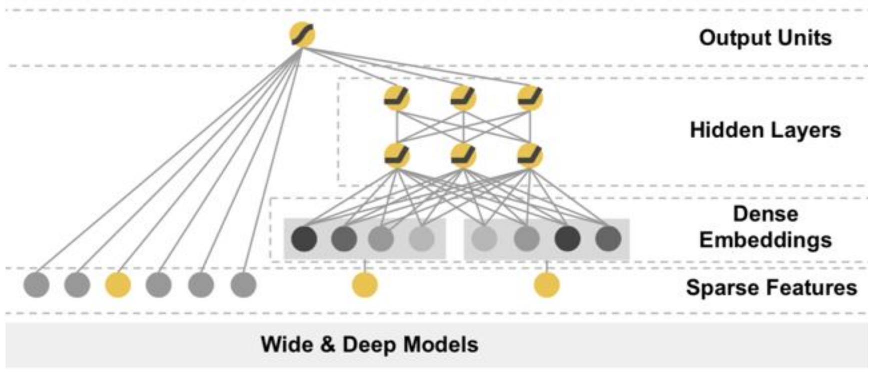


Figure1: The spectrum of Wide & Deep models.

1.1 Linear Regression Model

In 1991, Elton and Gruber assumed that the return of a security can be described by a multifactor linear model, and now linear regression approach has been widely used in the field of the stock market.

$$R = C + x_1F_1 + x_2F_2 + \ldots + x_nF_n \tag{1}$$

where R denotes the revenue of a specific stock, C denotes the constant of that linear model, and x and F denote stock factors and coefficient of that factor, respectively.

1.2 Wide and Deep Model

Wide and Deep Model is created by Google Inc. Here is a brief summary of the original paper explaining the wide and deep model. The wide and deep model we used is from tensorflow. See details in *Wide & Deep Learning for Recommender Systems* by Google Inc.

1.2.1 The Wide Component

The wide component of the Wide and Deep model is a generalized linear model of the form:

$$y = b + W^T x \tag{2}$$

where y is the responding variable. x is the feature vectors. b is the bias, and w is the parameter of each feature. A cross-product transformation is applied into the feature vectors, which is defined as:

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\} \tag{3}$$

where c_{ki} is a boolean variable that is 1 if the i-th feature is part of the k-th transformation ϕ k, and 0 otherwise. This transformation aims to find if there exist any interactions between each input variable.

1.2.2 The Deep Component

As it is shown in the right part of figure 1, to get the output units, the input sparse features will be going through two main stages. Sparse features are initially converted to a low-dimensional and real-value vector, which refers to as an embedding vector. After converting from sparse features into embedding vectors, these embedding vectors are then fed into the hidden layers of a neural network, where each hidden layer's computation is defined as:

$$a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}) \tag{4}$$

where l is the layer number and f is the activation function, often rectified linear units (ReLUs). $a^{(l)}$, $b^{(l)}$, and $W^{(l)}$ are the activations, bias, and model weights at l-th layer, respectively.

1.2.3 Joint Training of Wide & Deep Model

The wide component and deep component are combined by using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training.

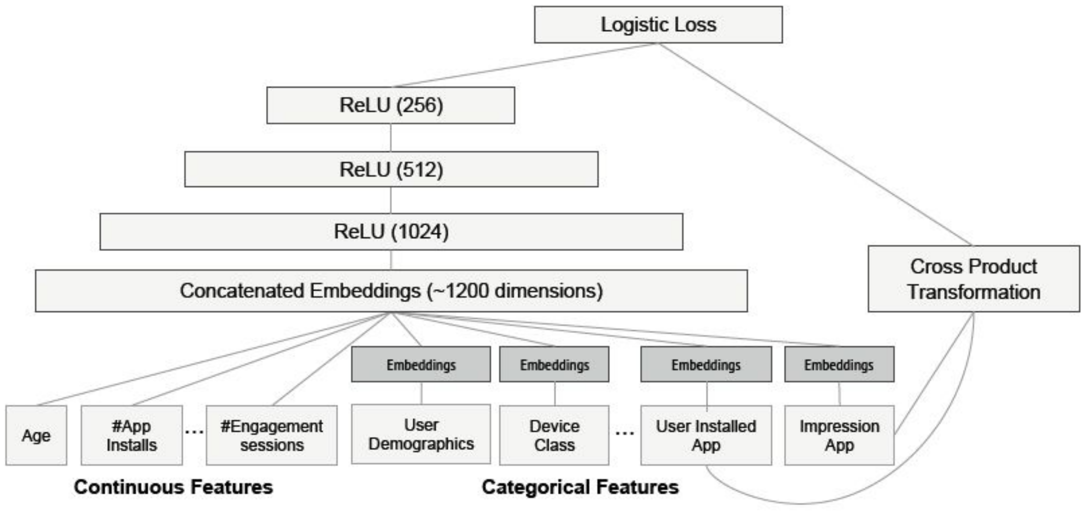


Figure 2: Wide & Deep model structure for apps recommendation

Joint training of a Wide & Deep Model is done by backpropagating the gradients from the output to both the wide and deep part of the model simultaneously using mini-batch stochastic optimization. Joint training optimizes all parameters simultaneously by taking both the wide and deep part as well as the weight of their sum into account at training time. We will be following the same process as the app recommender system do as shown in figure 2, where the original numerical variables along with embedded categorical vectors are replaced by the factors of stocks.

1.3 Stock selection

CSI 500 is an index conceived by China Securities Index Co .,Ltd. This index is derived by the top 500 ranking stocks according to their total market value. In this paper, we are going to make predictions and backtesting by analyzing the factors of these 500 stocks. Specifically, top 100 and last 100 ranked stocks will be predicted by their corresponding past factors, and a backtesting will then be applied on the top 100 and last 100 stocks.

2.DATA

2.1 Data & Factors

All data are accessed and modified with API on https://uqer.io/. See ‘uqer.py’ for details. Among the data, tradeDate is used for splitting the data into train, valid and test. label is the y value to be predicted, and the rests are the factors used in training.

1. **ticker**: stock code
2. **lastChgPct**, last month’s price change percentage

3. “**'01030101', '01030102', '01030103', '01030104', '01030105', '01030106', '01030107', '01030108', '01030109', '01030110', '01030111', '01030112', '01030113', '01030114', '01030116', '01030117', '01030118', '01030119', '01030301', '01030302', '01030303', '01030304', '01030305', '01030306', '01030307', '01030308', '01030309', '01030310', '01030311', '01030312', '01030313', '01030314', '01030315', '01030316', '01030317', '01030318', '01030319', '01030320', '01030321', '01030322', '01030323', '01030324', '01030325', '01030326', '01030327', '01030328', '01031400', '01031401', '01031402', '01031403', '01031404', '01031405', '01031406', '01031407', '01031408', '01031409'**”: Industry name indices
4. **ADX**: Average Directional Movement Index
5. **ASI**: Accumulation Swing Index
6. **FY12P**: Forecast earnings by analyst to market values
7. **Hurs**: Hurst exponent
8. **MACD**: Moving Average Convergence Divergence
9. **NetAssetGrowRate**: Net assets growth rate
10. **PB**: Price-to-book ratio
11. **PCF**: Price-to-cash-flow ratio
12. **PE**: Price-earnings ratio
13. **ROE**: Return on equity
14. **RSI**: Relative Strength Index
15. **VSTD10**: Volume Standard Deviation in 10 days
16. **industryID1**: Industry name
17. **marketValue**: Market value
18. **minusDI**: Decreasing component of DMI
19. **negMarketValue**: Circulation market value
20. **plusDI**: Increasing component of DMI
21. **exchangeCD**: Stock exchange name
22. **avgTurnoverRate**: Weekly turnover rate
23. **label**: stock price change percentage (responding variable)

2.2 Data Collection

For all of the trails below, the numerical features are normalized to [0,1] by using the formula $x = (x - x.min)/(x.max - x.min)$

-Trial 1

Factors: ticker, lastChgPct, marketValue, negMarketValue, PE, PB, FY12P, PCF, NetAssetGrowRate, VSTD10, avgTurnoverRate

Time: 20160101-20190630
Special data manipulation: drop all rows with empty entry (NaN)

-Trial 2
Factors: ticker, lastChgPct, marketValue, negMarketValue, PE, PB, FY12P, PCF, NetAssetGrowRate, VSTD10, MACD, RSI, ROE, plusDI, minusDI, ADX, ASI, Hurst, avgTurnoverRate,label
Time: 20160101-20190630
Special data manipulation: drop all rows with empty entry (NaN)

-Trial 3
Factors: All the factors mentioned in the last section.
Time: 20160101-20190630
Special data manipulation: drop all rows with more than 15 empty entry (NaN), and set the rest empty entrees to 0 after normalization

3. PROCEDURE

With the collected data, all processes are done on google colab. The wide and deep model we used is composed of *base_models.py, training_utils.py,* and *scoring_func.py* provided by Feiyang Pan. Some important parameters are:
'batchsize': 256,
'lr'(learning rate): 1e-4,
'emb_size'(embedding size): 256,
'n_epochs'(number of epochs): 4,
'n_hidden'(number of hidden layers): 3,
The procedure includes:

1. Preprocessing: Normalization of numerical columns and one-hot of catecorical .
2. Splitting the data by date: For each of the six tests on six months, the datasets include three parts: the tested month as “test”, 6 month before the tested month as “valid”, and 2.5 years before “valid” as “train”.
3. Modeling and training: Tensorflow is used for constructing the model

4. RESULT

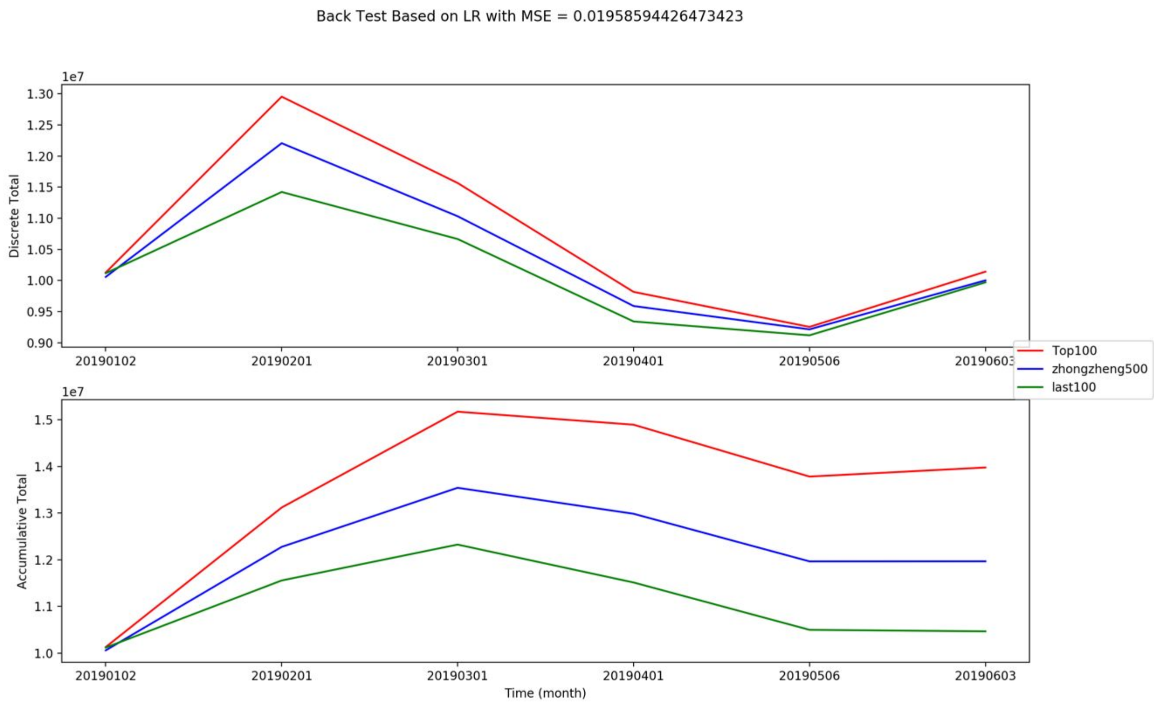
4.1 Benchmark for each trial

	# of Factors	MSE (LR)	MSE(widendeep)	Rev(LR)	Rev(widendeep)
Trial1	11	0.01958594	0.01864251	3978690.26	3653836.92
Trial2	20	0.01880872	0.01836933	3893489.29	3263673.82
Trial3	77	0.0210851	0.02965407	5045865.81	2073388.8

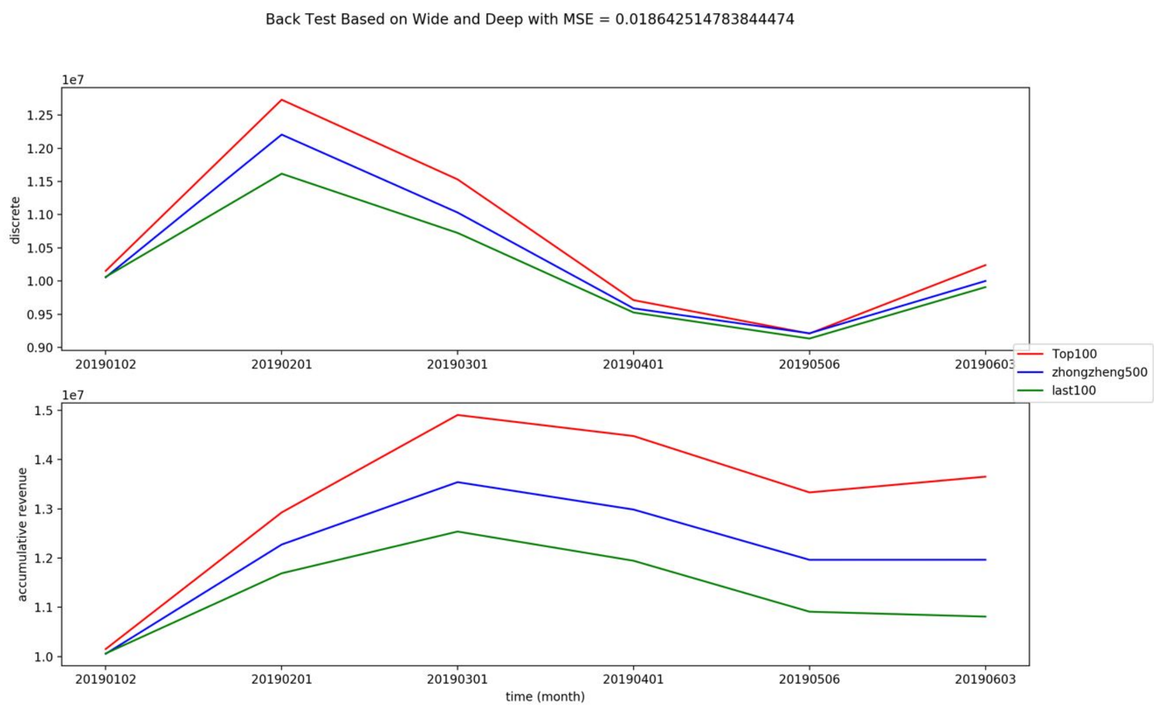
*Revenue (Rev) is based on initial 10000000 investment accumulatively.

4.2 Graph

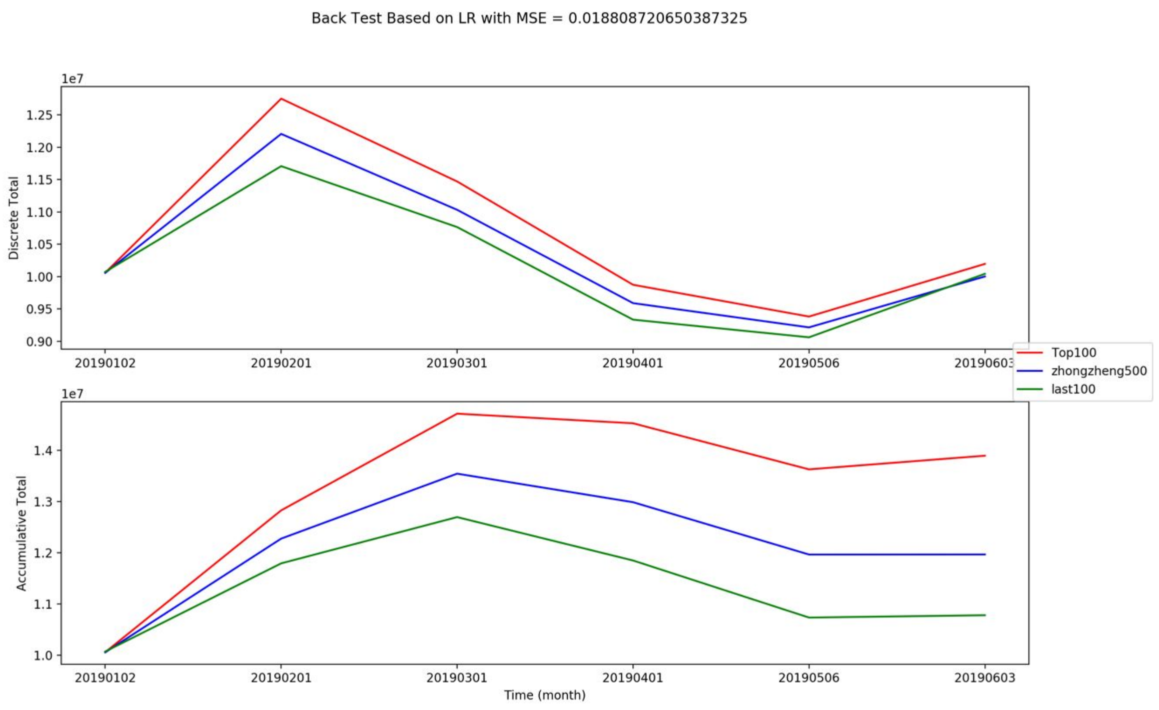
For the following graphs, the top graph is the result of investment with ¥ 10,000,000 per month. The bottom graph is the result of accumulative investment with a beginning fund of ¥10,000,000



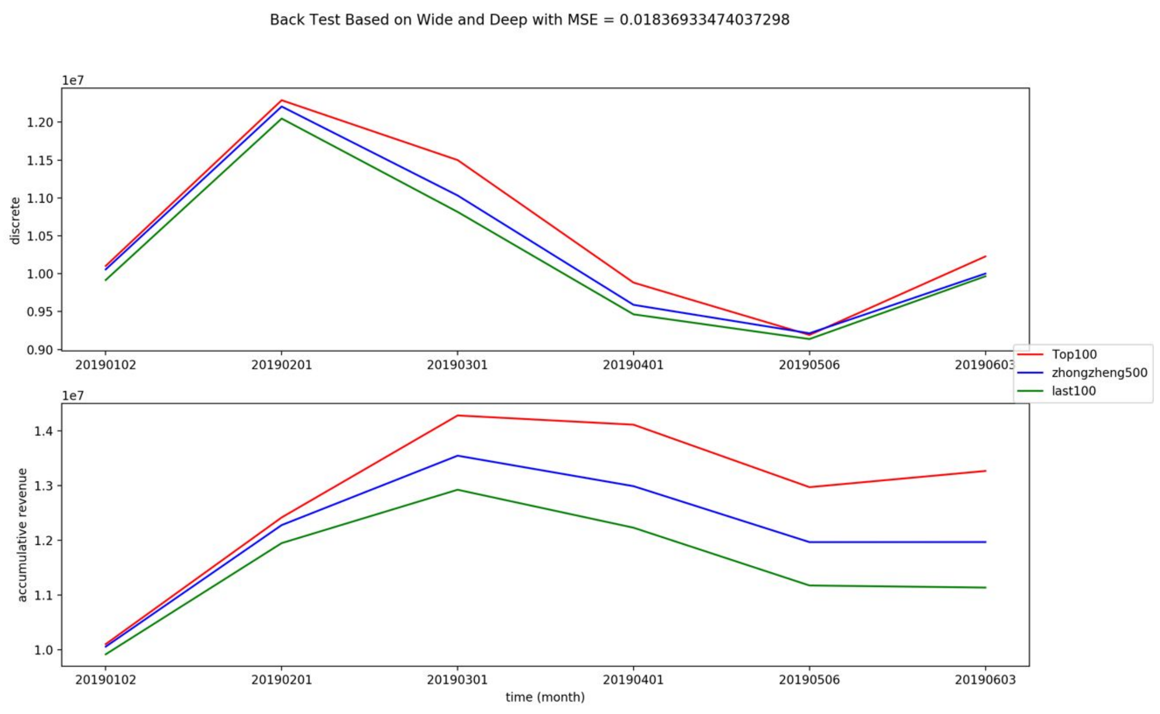
Linear Regression Trial 1



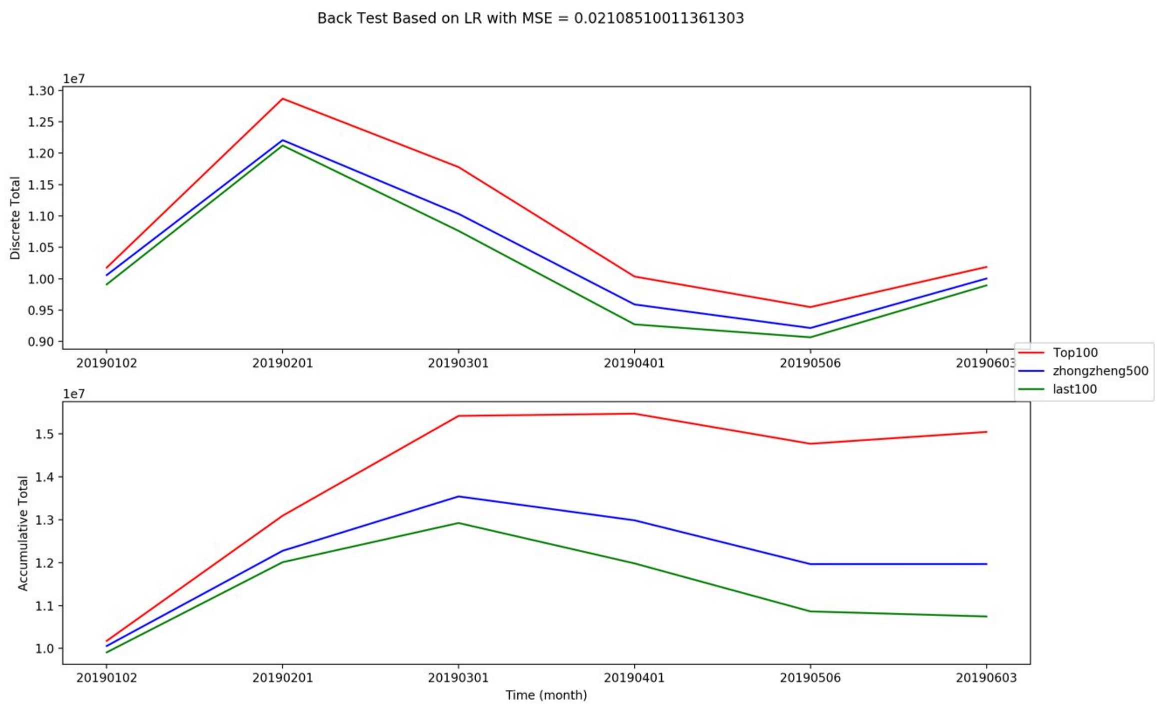
Wide and Deep learning Trial 1



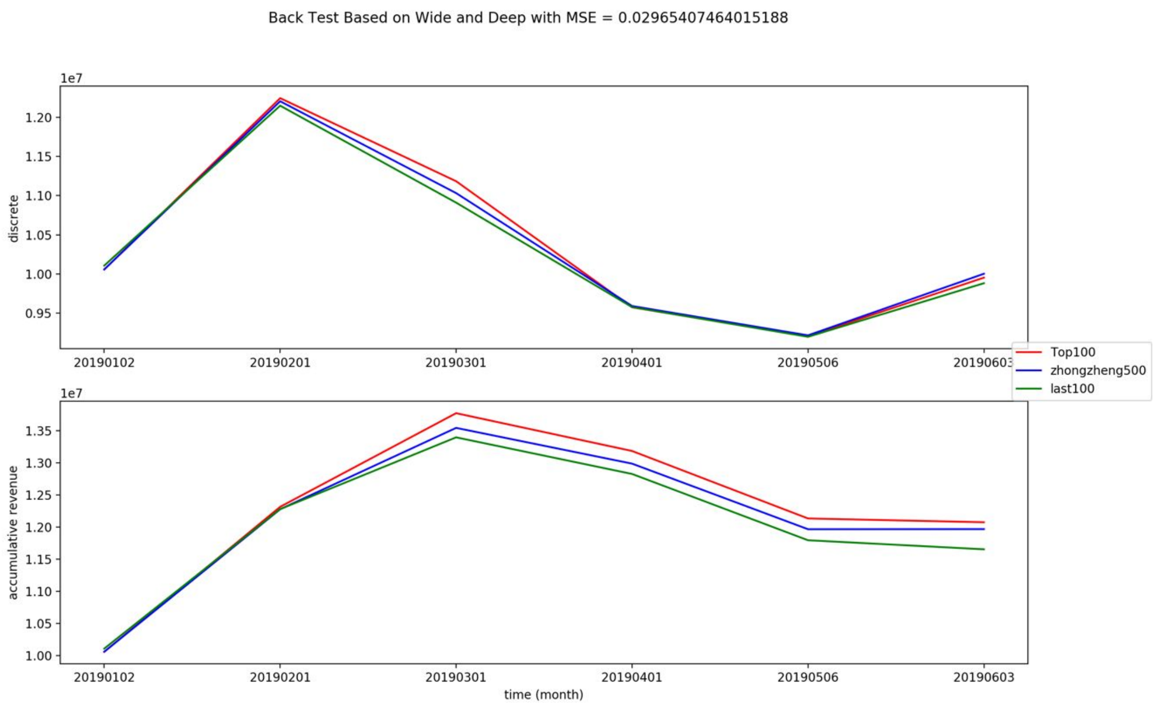
Linear Regression Trial 2



Wide and Deep learning Trial 2

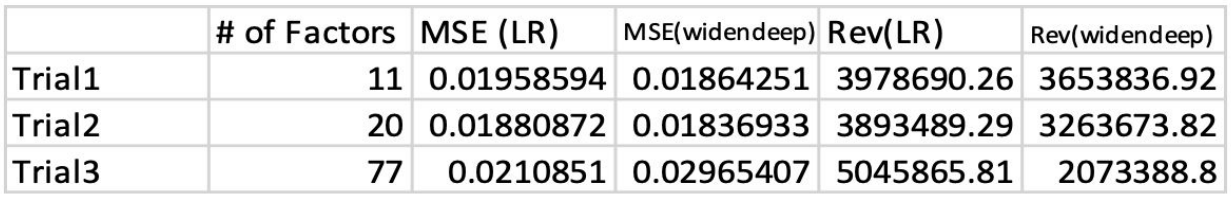


Linear Regression Trial 3



Wide and Deep Trial 3

5. ANALYSIS



	# of Factors	MSE (LR)	MSE(widendeep)	Rev(LR)	Rev(widendeep)
Trial1	11	0.01958594	0.01864251	3978690.26	3653836.92
Trial2	20	0.01880872	0.01836933	3893489.29	3263673.82
Trial3	77	0.0210851	0.02965407	5045865.81	2073388.8

While working through the trials, we have expected the performance of both LR and wide and deep to be better after learning from more factors. It turns out that although MSE goes down, the revenue given by wide and deep is as good as not as predicted. We contribute this to the randomness of wide and deep prediction. We also expected that after adding more categorical columns in trial 3, the performance of wide and deep should exceed that of LR. However, the result is unfortunately unexpected. The reason still need to be studied but I believe that it should be contributed to the dataset. The more than 50 categorical columns we add actually represent only one feature of the stock: the industry. To solve this problem, we need to either find a new way to represent several industry in one column, or look for other categorical columns.

From the table, we can also see that MSE has no direct relation with the revenue. This is because MSE only represent the accuracy of the predicted percentage price change of the stocks instead of the accuracy of final ranking based on the change. This is also the reason of the revenue earned by wide and deep not being as good as the revenue earned by LR even if it has higher accuracy. We choose to train the model to predict the actual price change because we only want to choose the top 100 stocks. An alternate way is to use rank as label, however, making the machine to predict the rank doesn't guarantee us to receive exactly 100 stocks. For further research and study, I suggest to try using rank as label and buy all stocks which are predicted to be the top rank in backtest. In this way, with relatively low MSE, wide and deep may give better result.

6. CONCLUSION

Although, according to Google's original paper, wide and deep model has been successful when being used in recommender systems, it does not hold its advantage in multi-factor stock selection. According to the overall result, wide and deep model is still effective in profiting, it does not show any improvement when comparing to the result given by LR model. We speculate the reason to be the simplicity of the factors we can accessed and the lack of categorical feature of the stocks compared to app acquisitions. Since almost all the features are numerical, the deep component and cross-product transformation of wide and deep model won't be able to show its advantage.